HCI プログラミング

12~14 回目　復習（基礎・応用・発展）　課題

■課題 1　**成績評価アプリを作成しましょう。得点をテキストフィールドに入力してリターンを押すと、評価（優／良／可／不可）が表示されます。得点と評価の対応、文字列の色は以下のように設定します。**

得点	評価	色
0 点～59 点	不可	赤〔Color.RED〕
60 点～69 点	可	黄〔Color.YELLOW〕
70 点～79 点	良	緑〔Color.GREEN〕
80 点～100 点	優	青〔Color.BLUE〕

レポートで同等のアプリを提出したら?

加点項目は以下のとおりで **40 点**（配点:基礎 70 点+完成度 10 点）の採点になります。

- レイアウト（+10 点）
- イベント処理（+10 点）
- テキストフィールドの利用（+10 点）
- 完成度（10 点）※入力に対する十分な例外処理、プロンプト文字列、ウィンドウサイズ変更などに起因するレイアウト乱れ防止など

その他の設定は次のとおりです。参考にしましょう。

レイアウト VBox の設定

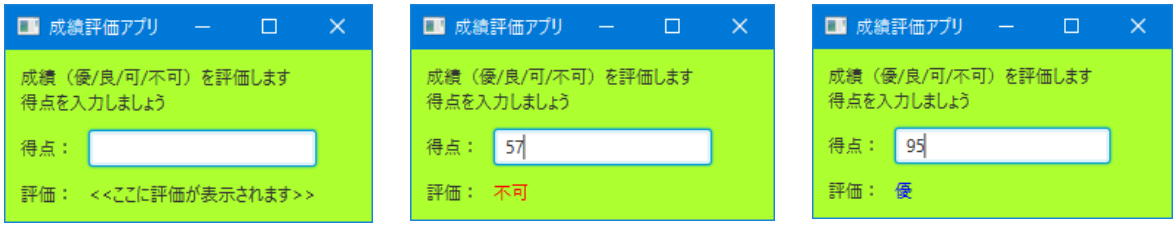
レイアウト周りの空白エリア　→　10 ピクセル〔setPadding(new Insets(10));〕
GUI 部品間の空白エリア　→　10 ピクセル〔setSpacing(10);〕

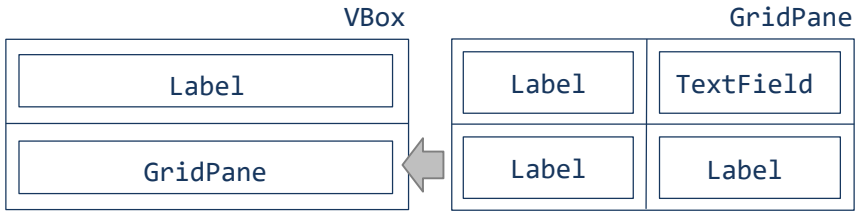
レイアウト GridPane の設定

GUI 部品間の空白エリア（縦）→　10 ピクセル〔setHgap(10);〕
GUI 部品間の空白エリア（横）→　10 ピクセル〔setVgap(10);〕
シーンの色　→　黄緑〔Color.GREENYELLOW〕
　※レイアウト VBox の背景色を透明にします
　　レイアウト VBox の setBackground(null);を実行
ウィンドウのタイトル　→　成績評価アプリ

ヒント 1：テキストフィールドにイベントハンドラを設定します。得点を入力してリターンキーが押されるとこのイベントハンドラが実行されます。
ヒント 2：イベントハンドラでは、テキストフィールドに入力された文字列を取得し、整数値に変換します。この整数値をもとに評価を行い、ラベルの文字列と色を変更します。

※おおよそ実行例のような画面になれば OK です
〔実行例〕





■課題 2　摂氏華氏変換アプリを作成しましょう。摂氏と華氏は次のように変換できます。

・摂氏 C→華氏 F　　F = 1.8*C+32
・華氏 F→摂氏 C　　C = (F-32)/1.8

メニューから変換の方向を選択し、テキストフィールドに変換元の温度を入力して変換ボタンを押します。変換された値が下のラベルに表示されます。

レポートで同等のアプリを提出したら?

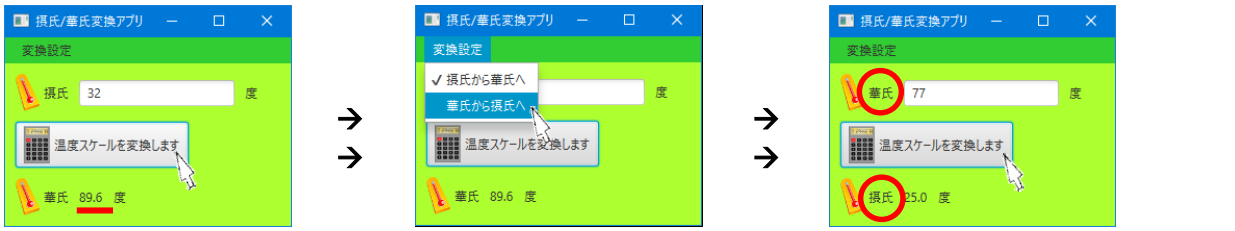
加点項目は以下のとおりで **70 点**（配点:基礎 70 点+完成度 10 点）の採点になります。
・イメージ（+10 点）・レイアウト（+10 点）・イベント処理（+10 点）・ボタン（+10 点）・メニュー（+10 点）・テキストフィールド（+10 点）・完成度（10 点）
※入力に対する十分な例外処理、プロンプト文字列、レイアウト乱れ防止、アクセラレータなど利便性など

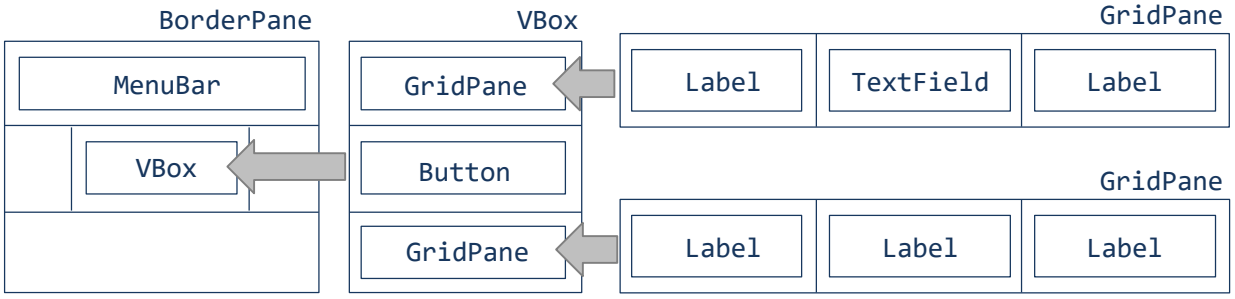
その他の設定は次のとおりです。参考にしましょう。

レイアウト VBox の設定
レイアウト周りの空白エリア　→　10 ピクセル〔setPadding(new Insets(10));〕
GUI 部品間の空白エリア　→　10 ピクセル〔setSpacing(10);〕
レイアウト GridPane の設定
GUI 部品間の空白エリア（横）→　10 ピクセル〔setHgap(10);〕
シーンの色　→　黄緑〔Color.GREENYELLOW〕
※レイアウト BorderPane の背景色を透明にします
このレイアウトの setBackground(null);を実行
メニューの背景色　→　ライムグリーン〔Color.LIMEGREEN〕
ウィンドウのタイトル　→　華氏／摂氏変換アプリ

ヒント 1：メニューとボタンに各々イベントハンドラ（別々のクラスで宣言）を設定します。
ヒント 2：予め変換の方向を表す変数をメンバー変数として宣言しておきましょう。例えば、boolean　Celsius2Fahrenheit;としてこれが true であれば摂氏から華氏へ変換のように。
ヒント 3：メニュー項目が選択されたら、この項目に応じて変数 Celsius2Fahrenheit を更新します。さらに、ラベルの「摂氏」と「華氏」を適切に入れ替えます。
ヒント 4：ボタンが押されたら、テキストフィールドに入力された文字列を取得し、実数値に変換します。この実数値を変数 Celsius2Fahrenheit に従った変換方向で計算します。最後に、変換結果をラベルの文字列に設定して表示します。

※画像は HP よりダウンロードしてソースファイルと同じフォルダに保存しておきましょう
※おおよそ実行例のような画面になれば OK です
〔実行例〕

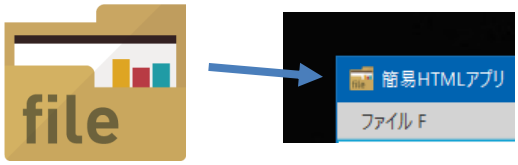




■課題 3　クラス HTMLEditor を用いて簡易 HTML エディタを作成してみましょう。クラス HTMLEditor は HTML ファイルの表示と編集を支援します。メニューを下記のように構成し，ファイルの読込と保存、印刷の機能を加えましょう。さらに、アプリアイコン（タイトルバー左隅に表示されるアイコン）を設定してみましょう。

メニュー構成：
[ファイル]-[開く][保存]|[印刷]|[終了]

アプリアイコン：



レポートで同等のアプリを提出したら?

加点項目は以下のとおりで **50 点**（配点:基礎 70 点+完成度 10 点）の採点になります。
・イメージ（+10 点）・レイアウト（+10 点）・イベント処理（+10 点）・メニュー（+10 点）・完成度（10 点）
※例外への対応、レイアウト乱れ防止、アクセラレータなど利便性など

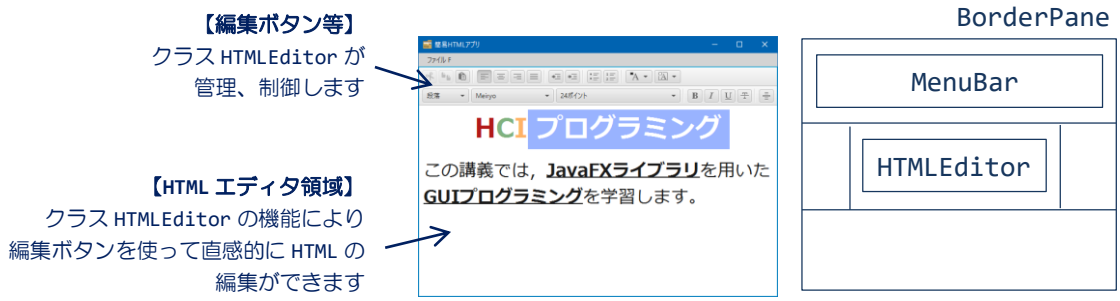
その他の設定は次のとおりです。参考にしましょう。
ウィンドウのタイトル　　→　簡易 HTML アプリ

ヒント 1：アプリアイコンは Stage クラスの getIcons()メソッドで現在のアイコンリスト（ObservableList<Image>型）を取得することができます。メニュー項目と同様に add()メソッドで、画像ファイルに関連付いた Image クラスのオブジェクトを追加します。
ヒント 2：読込みファイルの選択には、クラス FileChoose の showOpenDialog()メソッドを、保存ファイルの選択には、同クラスの showSaveDialog()メソッドを用いましょう。対象ファイルからの入出力は BufferedReader クラスと PrintWriter クラスを利用できます。
ヒント 3：編集データの印刷には、クラス FileChoose の print()メソッドを用いましょう。

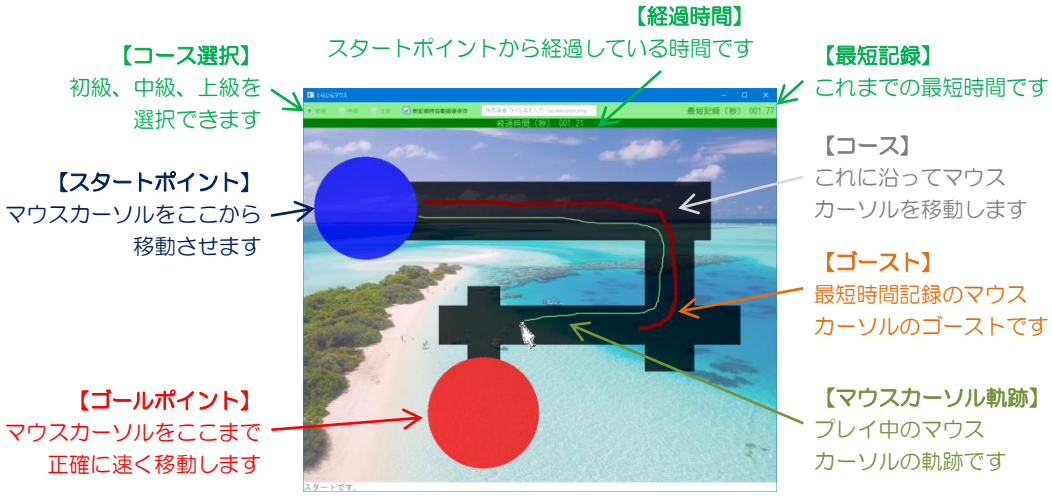
※クラス HTMLEditor 利用、ファイル入出力、印刷機能には次の import 文を加えましょう。さらに、実行時にクラス HTMLEditor が利用できるようランタイム部品（モジュール）を追加（javafx.web モジュールの追加読込）するため tasks.json（HP の課題 3 付近に掲載）を更新しましょう。

```
import javafx.scene.web.*;
import java.io.*;
import javafx.print.*;
```

※画像は HP よりダウンロードしてソースファイルと同じフォルダに保存しておきましょう
※おおよそ実行例のような画面になれば OK です
〔実行例〕

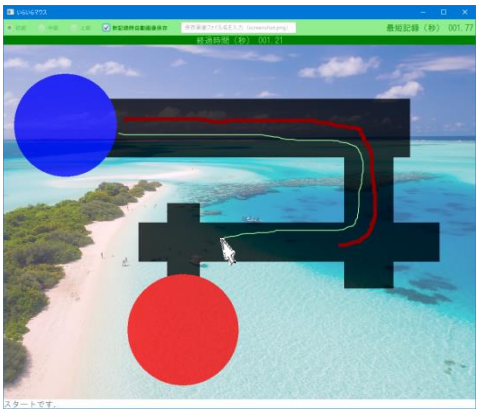


■課題 4　マウスを用いたタイムトライアルゲーム「いらいらマウス」を作成しましょう。
ゲーム内容は以下の通りです。



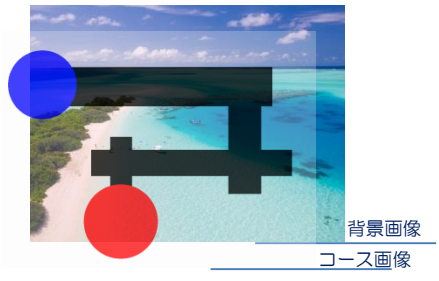
〔ゲームの内容〕
マウスカーソルを青色の円で示されるスタートポイントから黒色で示されるコースを通りぬけ、出来るだけ速く赤色の円で示されるゴールポイントまで移動させます。マウスカーソルを移動している間はリアルタイムで経過した時間が表示されます。同時にこれまでの最短記録でのマウスカーソルの軌跡がゴーストとして表示されます。最短記録を更新したら、今回の新記録が保存され、そのマウスカーソルの軌跡がゴーストになります。また、難易度の異なる３種類のコースの変更や新記録達成時のスクリーンショット保存の機能があります。
★発展課題の達成は**「スタートポイントからゴールポイントまでの移動に要する経過時間の計測とその結果の表示」**とします。

〔画面の設計〕



〔キャンバスの構成とコース画像の色設計〕
キャンバス上にコースを配置します。2 枚の画像（背景画像とコース画像）を重ね描画します。コース画像は半透明処理済みで背景が透けます。コース画像の色設計です。

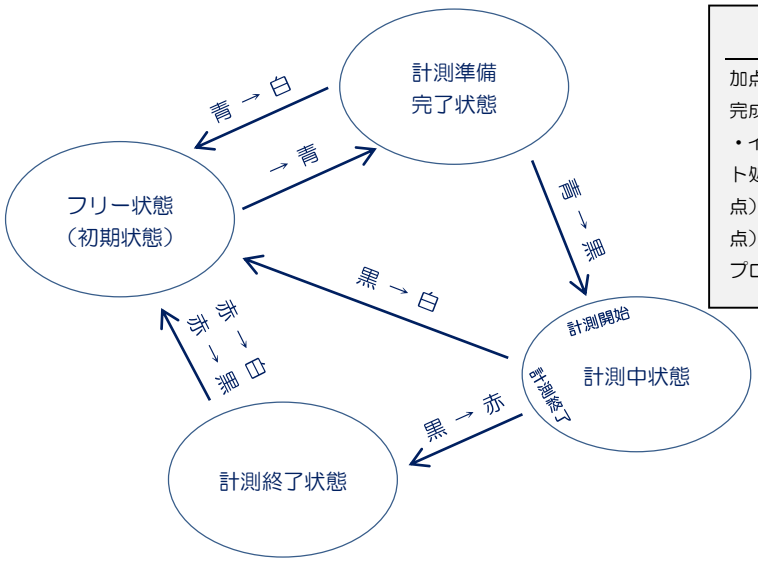
〔スタート領域〕　青 RGB 0.0, 0.0, 1.0
〔ゴール領域〕　赤 RGB 1.0, 0.0, 0.0
〔コース領域〕　黒 RBG 0.0, 0.0, 0.0
〔上記以外の領域〕　白 RBG 1.0, 1.0, 1.0



〔マウスカーソルの位置する領域の判断〕
コース画像にはスタート領域とゴール領域、コース領域、それ以外の領域があります。現在のマウスカーソルの位置がどの領域に属しているかは、そのカーソル位置の画素情報を取得して上記の色設計をもとに判断します。画像が格納されている Image クラスを用いて以下のように画素情報を得ることができます。現在のマウスカーソル位置を(mx, my)とします。

```
Image im = new Image("Sample.png");
PixelReader pl = im.getPixelReader();
Color c = pl.getColor((int)mx, (int)my);
double r = c.getRed();   // カーソル位置(mx,my)の画素値（赤）0.0 ~ 1.0
double g = c.getGreen(); // カーソル位置(mx,my)の画素値（緑）0.0 ~ 1.0
double b = c.getBlue();  // カーソル位置(mx,my)の画素値（青）0.0 ~ 1.0
```

〔ゲームの状態遷移〕
マウスカーソルは上記の 4 領域（以下では、4 領域の対応色である青、赤、黒と白で表現します）を行き来します。この情報をもとに、ゲームをスタートする（計測を開始する）タイミングとゴールをした（計測を終了する）タイミングを判断します。
　以下のように 4 つのゲームの状態を考えて計測のタイミングをとります。



レポートで同等のアプリを提出したら?

加点項目は以下のとおりで **80 点**（配点:基礎 70 点+完成度 10 点）の採点になります。
・イメージ（+10 点）・レイアウト（+10 点）・イベント処理（+10 点）・ボタン（+10 点）・メニュー（+10 点）・テキストフィールド（+10 点）・キャンバス（+10 点）・完成度（10 点）※入力に対する十分な例外処理、プロンプト文字列、レイアウト乱れ防止など

ゲームを起動したとき、最初のゲームの状態は「フリー状態」です。この「フリー状態」ではマウスカーソルはどこの領域に移動しても計測は開始しません。計測を開始するには、青領域にマウスカーソルを移動して、ゲームの状態を「計測準備完了状態」にします。次に、青領域から黒領域にマウスカーソルを移動すると「計測中状態」となり、このタイミングで計測を開始します。無事に黒領域からはみ出ずに赤領域まで移動すると、ゲームの状態は「計測終了状態」になり、このタイミングで計測を終了します。その後、赤領域から白領域や黒領域に出ると、「フリー状態」にもどります。もし「計測中状態」でマウスカーソルが黒領域から白領域にはみ出した場合も「フリー状態」にもどります。

〔経過時間の取得〕
スタートポイントからの経過時間は協定世界時 1970.1.1 からの経過時間をミリ秒で求める System クラスの long currentTimeMillis()メソッドを用います。スタート時にこの値 s を保存しておき、ゴール中に再度この値 t を求めて t-s で経過時間を計算します。